Creating Interoperable Meshing and Discretization Technology:

The Terascale Simulation Tools and Technologies Center

David Brown *(Lawrence Livermore)*

Lori Freitag *(Argonne)*

Jim Glimm *(Brookhaven, SUNY Stony Brook)*

Presented at the Computational Fields Simulation Conference

June 3, 2002

Terascale Simulation Tools and Techology

TSTT

UCRL-PRES-151494

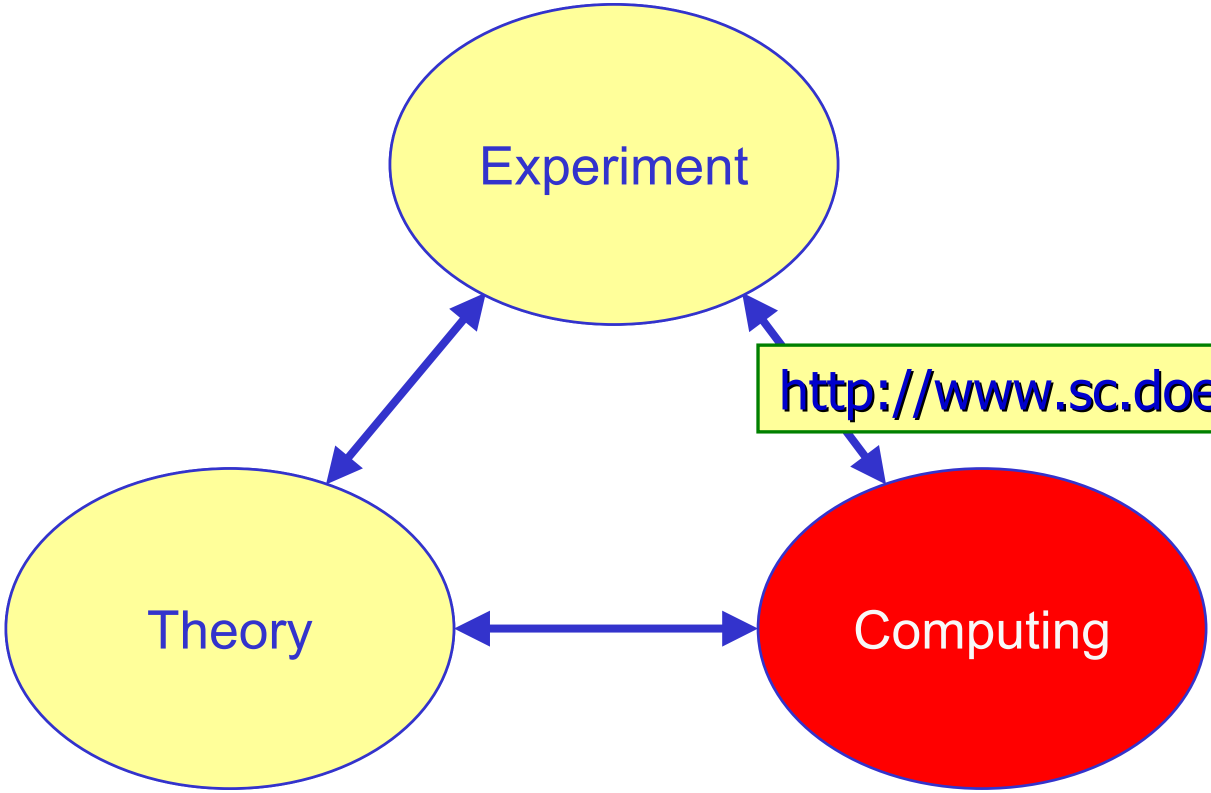
This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

TSTT is a SciDAC Integrated Scientific Infrastructure Center ← "ISIC"

SciDAC
Scientific Discovery through Advanced Computing

- SciDAC is a 5-year program in the Office of Science at the Department of Energy
- Develop Scientific Computing Software and Hardware Infrastructure needed to use terascale computers to advance DOE research programs
  - basic energy sciences
  - biological and environmental research
  - fusion energy sciences
  - high-energy and nuclear physics

The SciDAC vision enables new scientific discoveries by adding computing to theory and experiment



http://www.sc.doe.gov/scidac/

Leads to greatly enhanced cycle of understanding and innovation

The Applied Mathematics ISICs will provide scalable numerical libraries

Algorithmic and Software Framework for Applied Partial Differential Equations (APDEC) **AMR**	LLNL NYU U. of Washington U. of Calif. Davis U. of Wisconsin U. of North Carolina (Phil Colella, LBNL)	$2.7M/Yr
Terascale Optimal PDE Simulations (TOPS) **Solvers; optimization**	LLNL ANL LBNL Carnegie Mellon U. New York U. U. of Calif. Berkeley U. of Colorado U. of Tennessee (David Keyes, Old Dominion U.)	$3.3M/Yr
Terascale Simulation Tools & Technologies Center (TSTT)	ANL LLNL ORNL PNNL SNL RPI SUNY Stony Brook (Jim Glimm, BNL; David Brown LLNL; Lori Freitag ANL)	$2.6M/Yr

The Applied Mathematics ISICs will provide scalable numerical libraries

Algorithmic and Software Framework for Applied Partial Differential Equations (APDEC) **AMR**	LLNL NYU U. of Washington U. of Calif. Davis U. of Wisconsin U. of North Carolina (Phil Colella, LBNL)	$2.7M/Yr
Terascale Optimal PDE Simulations (TOPS) **Solvers; optimization**	LLNL ANL LBNL Carnegie Mellon U. New York U. U. of Calif. Berkeley U. of Colorado U. of Tennessee (David Keyes, Old Dominion U.)	$3.3M/Yr
Terascale Simulation Tools & Technologies Center (TSTT) **Meshes and discretization**	ANL LLNL ORNL PNNL SNL RPI SUNY Stony Brook (Jim Glimm, BNL; David Brown LLNL; Lori Freitag ANL)	$2.6M/Yr

The Computer Science ISICs will develop systems software and tools for terascale computers

Four activities focused on a comprehensive, portable, and fully integrated suite of systems software and tools for effective utilization of terascale computers

Scalable Tools for Large Clusters; Resour Interfa Framework **Parallel system tools**	ANL LBNL Ames PNNL SNL LANL National Center for Supercomputing App (Al Giest, ORNL)	$2.2M/Yr
High-End Computer Sysetms Performance: Science & Engineering **Performance**	ANL LLNL ORNL U. of Illinois UCSD U. of Tennessee U. of Maryland (David Bailey, LBNL)	$2.4M
Center for Component Technology for Terascale Simulation Software	ANL LANL LLNL ORNL PNNL U. of Utah Indiana U. (Rob Armstrong, SNL)	$3.1M
Scientific Data Management Enabling Technology Center **Data Management**	ANL LLNL ORNL Georgia Tech. UCSD Northwestern U. North Carolina State (Ari Shoshani, LBNL)	$3M

The Computer Science ISICs will develop systems software and tools for terascale computers

Four activities focused on a comprehensive, portable, and fully integrated suite of systems software and tools for effective utilization of terascale computers

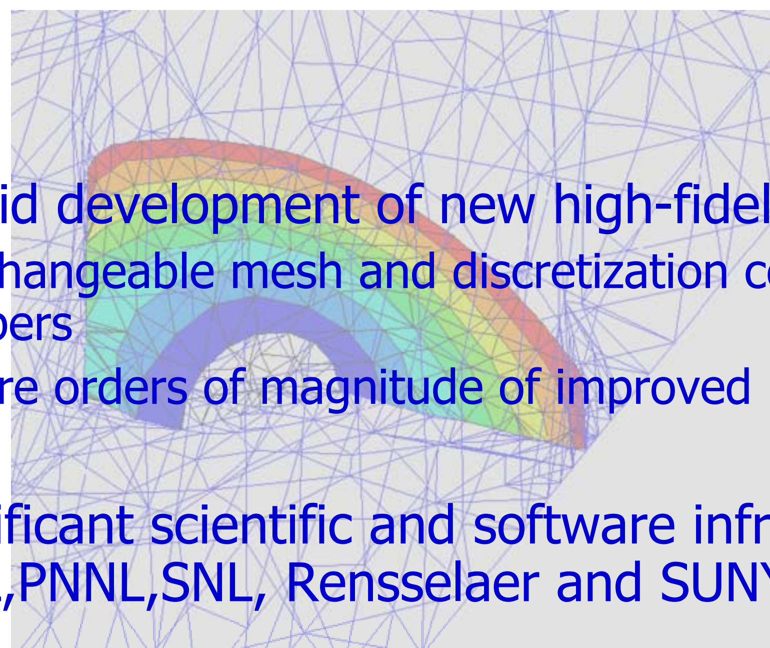
Scalable Tools for Large Clusters; Resour Interfa Framework **Parallel system tools**	ANL LBNL Ames PNNL SNL LANL National Center for Supercomputing App (Al Giest, ORNL)	$2.2M/Yr
High-End Computer Sysetms Performance: Science & Engineering **Performance**	ANL LLNL ORNL U. of Illinois UCSD U. of Tennessee U. of Maryland (David Bailey, LBNL)	$2.4M
Center for Component Technology for Terascale Simulation Software **Scientific software components**	ANL LANL LLNL ORNL PNNL U. of Utah Indiana U. b Armstrong, SNL)	$3.1M
Scientific Data Management Enabling Technology Center **Data Management**	ANL LLNL ORNL Georgia Tech. UCSD Northwestern U. North Carolina State (Ari Shoshani, LBNL)	$3M

http://www.sc.doe.gov/scidac/

Terascale Simulation Tools and Techology

TSTT

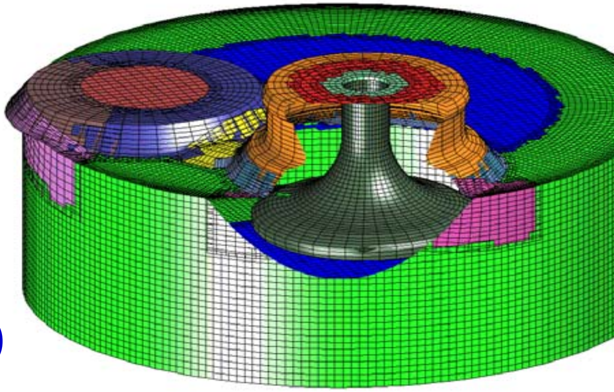
- TSTT is developing novel mesh and discretization tools
  - to allow modular plug-and-play technology insertion
  - to provide novel high-order and adaptive refinement tools
- TSTT will provide enhanced resolution/efficiency for 5 existing SciDAC application areas by inserting advanced meshing technology
  - *fusion*
  - *climate*
  - *accelerator design*
  - *combustion chemistry*
  - *supernova models*



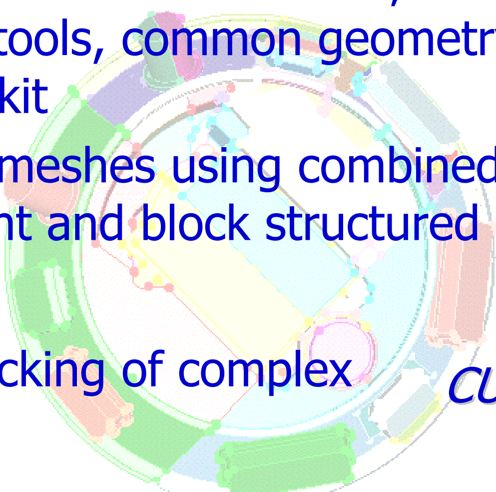
- TSTT will enable rapid development of new high-fidelity physics software
  - by providing interchangeable mesh and discretization components to application developers
  - yielding one or more orders of magnitude of improved physics resolution
- TSTT leverages significant scientific and software infrastructure from ANL,BNL,LLNL,ORNL,PNNL,SNL, Rensselaer and SUNY Stony Brook

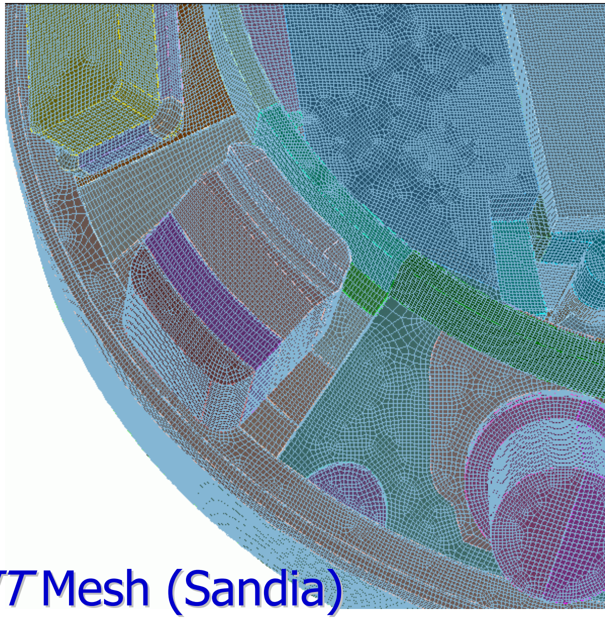
TSTT brings together existing meshing and discretization expertise from DOE Labs and Universities

- Structured and mixed-element meshes
  - **Overture** - high quality predominantly structured meshes on complex CAD geometries, mesh refinement (LLNL)
  - Variational and Elliptic Grid Generators (ORNL, SNL)
- Unstructured meshes
  - **MEGA** (RPI) - primarily tetrahedral meshes, boundary layer mesh generation, curved elements, mesh refinement
  - **CUBIT** (SNL) - primarily hexahedral meshes, automatic decomposition tools, common geometry module, mesh quality toolkit
  - **NWGrid** (PNNL) - hybrid meshes using combined Delaunay, mesh refinement and block structured
- Front-tracking
  - **FronTier** (SUNY-SB) - tracking of complex interfaces



Overture Mesh (LLNL)





CUBIT Mesh (Sandia)

Bringing this sophisticated technology to DOE application scientists is the challenge

- DOE has supported R&D leading to sophisticated tools for
  - structured, unstructured, hybrid mesh generation
  - front-tracking, local mesh refinement
  - high-order PDE discretization methods
- In general, however, the technology requires too much software expertise from application scientists
  - Difficult to improve existing codes
  - Difficult to design and implement new codes

We meet this challenge through a 2-pronged approach

Near term collaborations helps us understand application requirements …

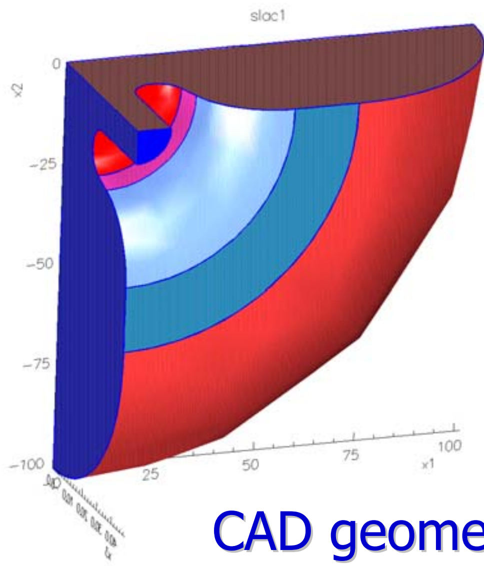
- *Near term:* deployment of current TSTT mesh and discretization capabilities by partnering with SciDAC applications
- *Long term:* development of interoperable software tools enabling
  - Rapid prototyping of new applications
  - Plug-and-play insertion of mesh and discretization technology through uniform software interfaces

… feeding into interface design of future software components

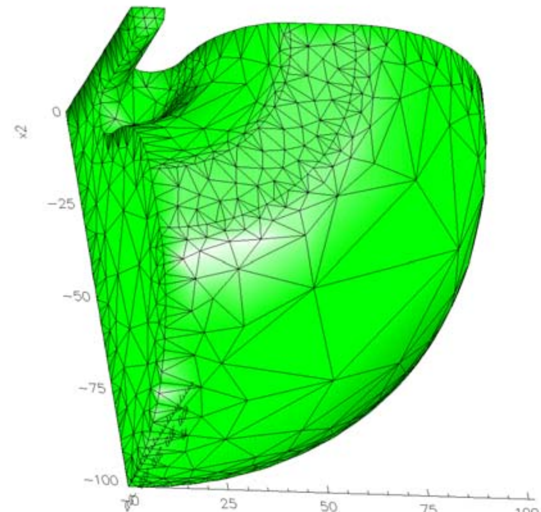
TSTT technology is helping near-term SciDAC application progress

- MESQUITE mesh quality methods are being used to understand E&M code sensitivity to mesh quality (SLAC)
- Analysis of time-domain FD methods to improve stability of accelerator E&M code (SLAC)
- Higher-order and adaptive FEM solution strategies are under study for PPPL par-M3D fusion code (Princeton)
- Spectral Elements for Climate Dynamical Core (NCAR)
- Front tracking for modeling of the breakup of a diesel fuel jet into spray (Argonne)
- Adaptive quadrature method for discrete ordinates method for Boltzman transport equations in supernova application (ORNL)

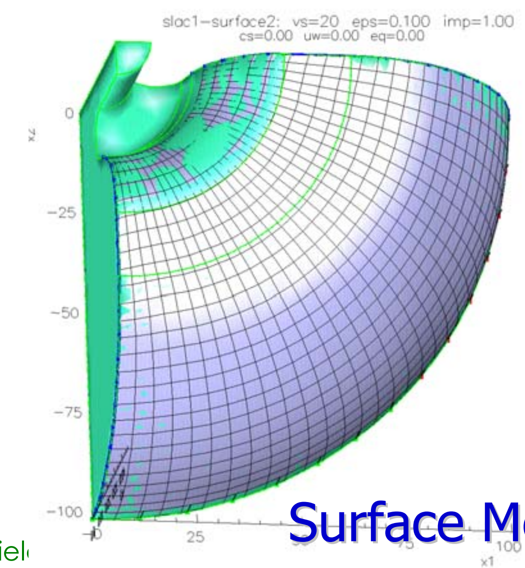
Overture/Rapsodi tools used to mesh accelerator geometries for SLAC



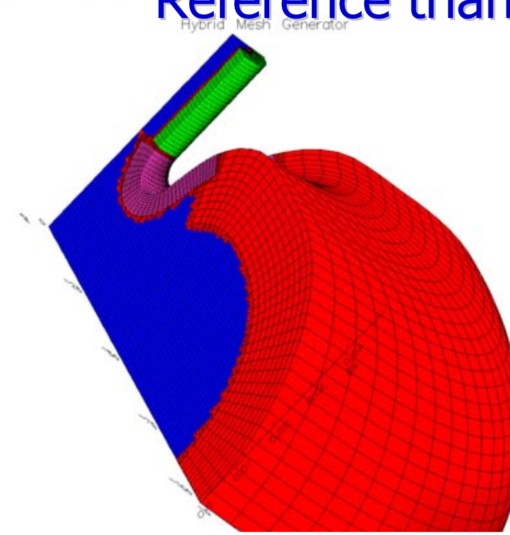
CAD geometry



Reference triangulation



Surface Mesh



Hybrid volume mesh

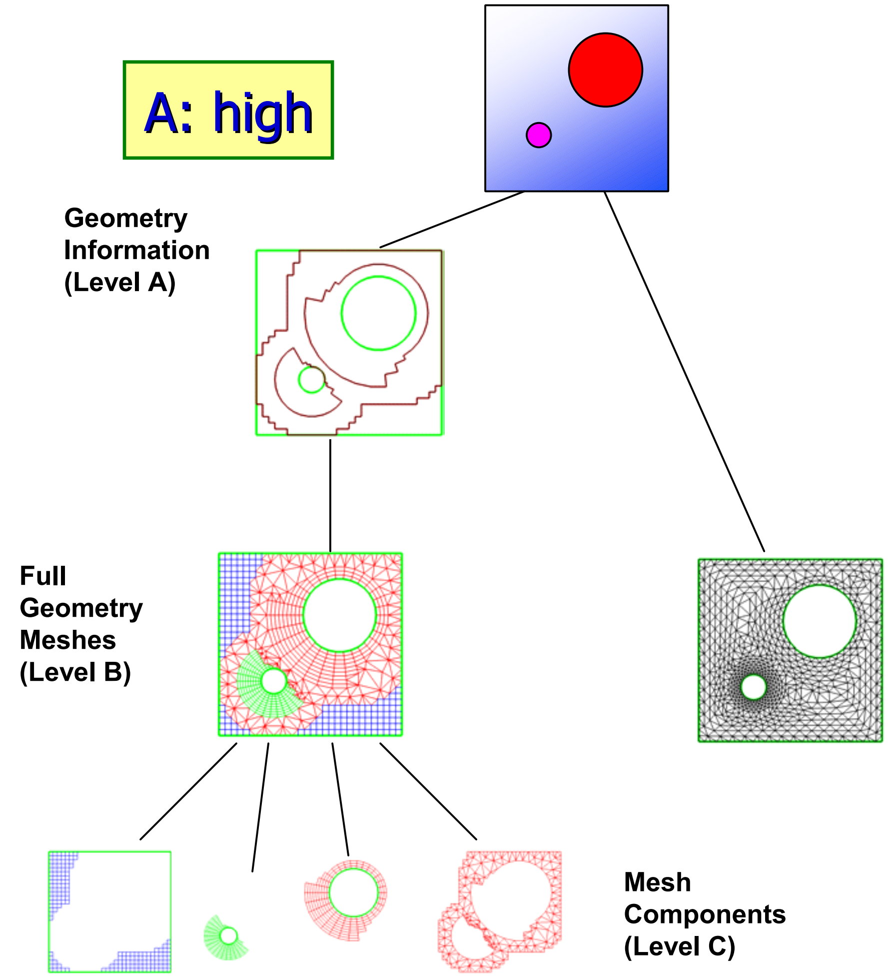
Terascale Simulation Tools and Techology

While there are many well-developed technologies for simulation; they cannot be used interchangeably

- Many mesh types
  - Structured, unstructured, overlapping, mixed element, tetrahedral, hexahedral…
- Many discretization approaches
  - Finite difference, finite volume, finite element, spectral element, discontinuous Galerkin
- But the fundamental concepts for simulations are the same:
  - *Represent geometry with a mesh*
  - *Approximate functions and operators on the mesh*
  - *Adaptive mesh refinement; moving geometry; data transfer between meshes*
- TSTT Goal: Develop a common interface to multiple meshing and discretization technologies

Access to geometrical information at different levels of abstraction through *Mesh Data Hierarchy*

- Level A: Geometric description of the domain
  - provides a common frame of reference for all tools
  - facilitates multilevel solvers
  - facilitates transfer of information in discretizations
- Level B: Full geometry hybrid meshes
  - mesh components
  - communication mechanisms that link them (key new research area)
  - allows structured and unstructured meshes to be combined in a single computation
- Level C: Mesh Components



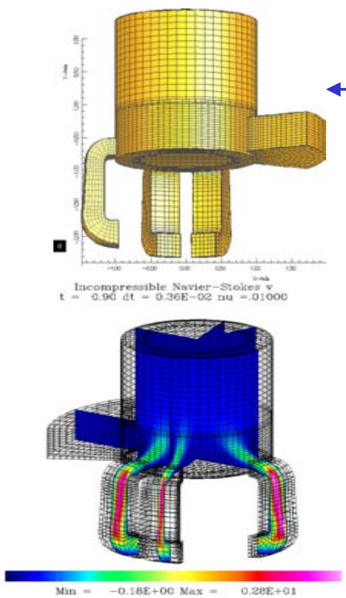
C: low

TSTT will provide powerful tools quickly by wrapping existing technology with a common interface specification

- Initial focus is on low level access to static mesh components
  - Data: mesh geometry, topology, field data
  - Avoid potential inefficiency through
    - Access patterns appropriate for each mesh type
    - Caching strategies and agglomerated access
    - Interaction with Performance ISIC
  - Appropriateness through working with
    - Application scientists
    - TOPS, CCA and APDEC SciDAC Math and CS Centers
  - "Plug-and-play": Application scientists program to the common interface and can than use any conforming tool without changing their code
- High level interfaces
  - to entire grid hierarchy which allows interoperable meshing by creating a common view of geometry
  - mesh refinement including error estimators and curved elements
- All TSTT tools will be CCA interface compliant

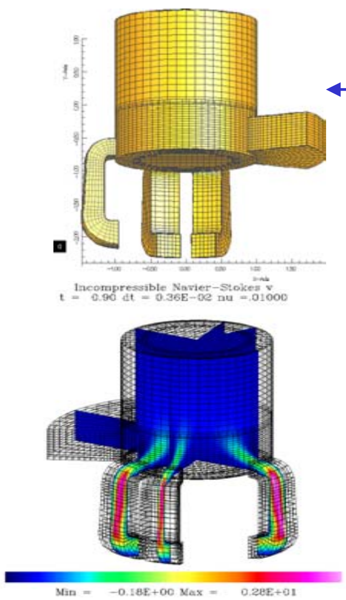
Rapid prototyping of new applications enabled by high-level access to mesh and data objects

Overture (LLNL) prototype




```
CompositeGrid cg;
floatCompositeGridFunction u,v,w;
```

Rapid prototyping of new applications enabled by high-level access to mesh and data objects



Overture (LLNL) prototype

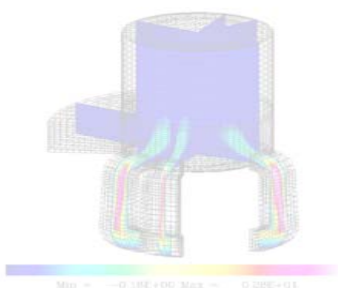
```
CompositeGrid cg;
floatCompositeGridFunction u,v,w;
```

Differentiation Operators

```
v = u.y();
w = u.laplacian();
```

Rapid prototyping of new applications enabled by high-level access to mesh and data objects

Overture (LLNL) prototype



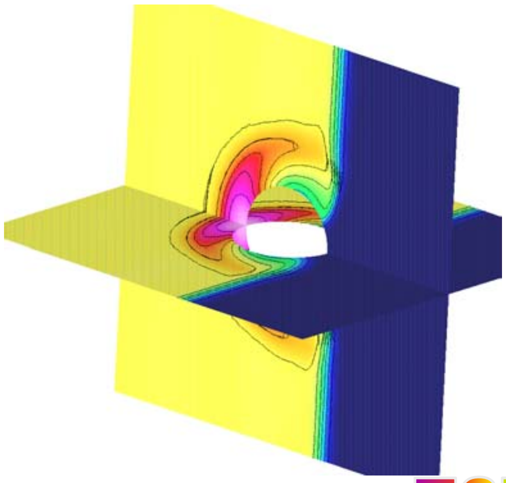
```
CompositeGrid cg;
floatCompositeGridFunction u,v,w;
```

Differentiation Operators

```
v = u.y();
w = u.laplacian();
```

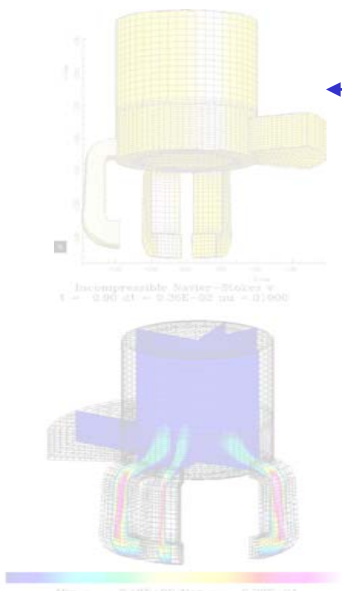
Visualize grid and data

```
Plotstuff ps;
ps.plot (cg);
ps.contour (w);
```



Rapid prototyping of new applications enabled by high-level access to mesh and data objects

Overture (LLNL) prototype



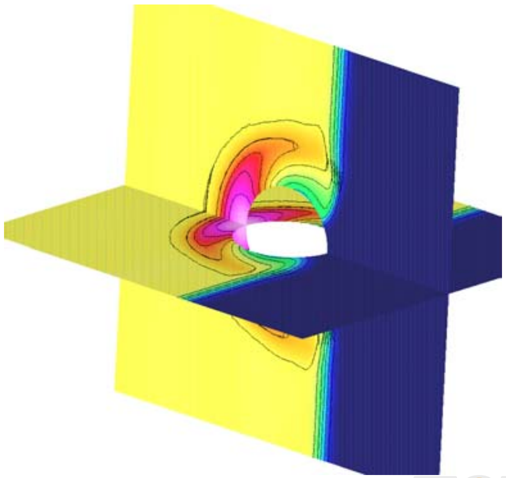
```
CompositeGrid cg;
floatCompositeGridFunction u,v,w;
```

Differentiation Operators

```
v = u.y();
w = u.laplacian();
```

Visualize grid and data

```
Plotstuff ps;
ps.plot (cg);
ps.contour (w);
```



Trellis (RPI) provides similar capability for finite-element method

High-order discretization methods can deliver improved accuracy with fewer degrees of freedom

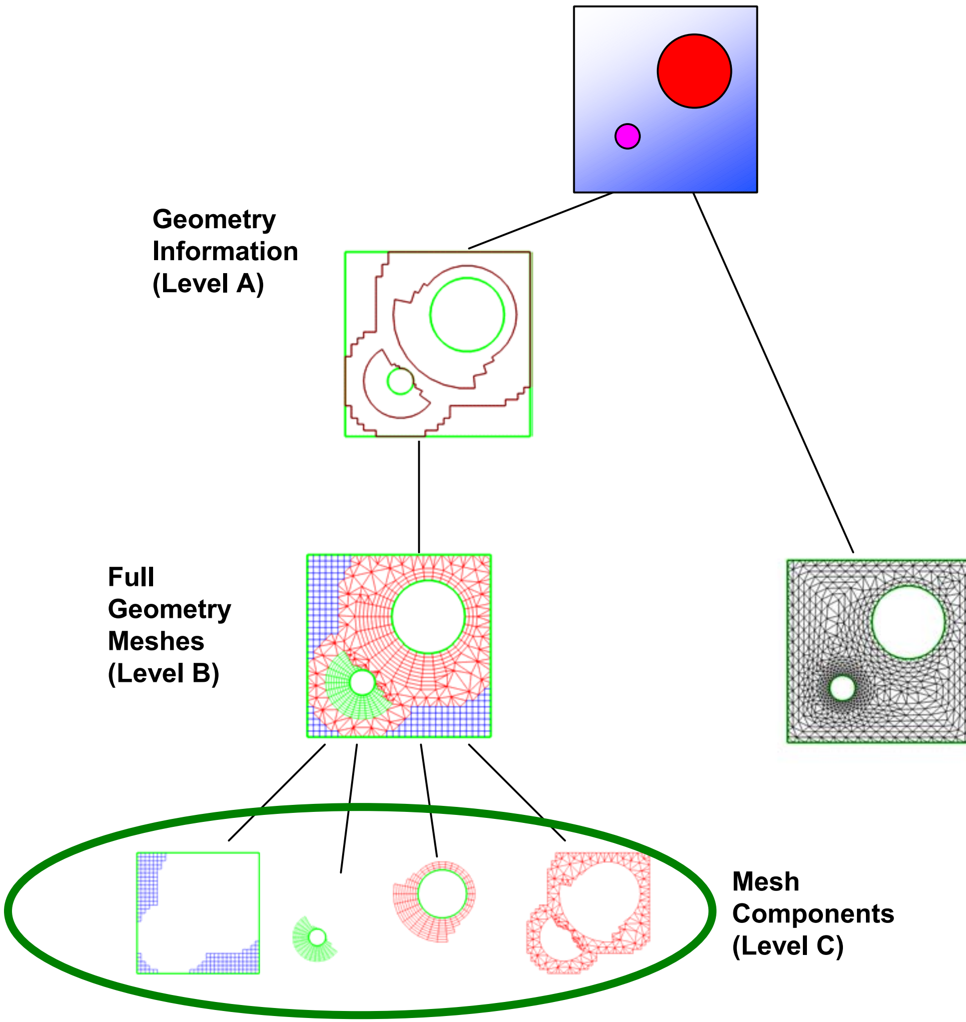
- However, complexities of using high-order methods on adaptively evolving grids has hampered their widespread use
  - Tedious low level dependence on grid infrastructure
  - A source of subtle bugs during development
  - Bottleneck to interoperability of applications with different discretization strategies
  - Difficult to implement in general way while maintaining optimal performance
- Result has been a use of sub-optimal strategies or lengthy implementation periods
- TSTT will eliminate these barriers by developing a *Discretization Library*

The TSTT discretization library will leverage similar work by the Overture and Trellis projects

- Mathematical operators will be implemented
  - Start with +, -, *, /, interpolation, prologation
  - Move to div, grad, curl, etc.
  - Both strong and weak (variational) forms of operators when applicable
- Many discretization strategies will be available
  - Finite Difference, Finite Volume, Finite Element, Discontinuous Galerkin, Spectral Element, Partition of Unity
  - Emphasize high-order and variable-order methods
  - Extensive library of boundary condition operators
- The interface will be independent of the underlying mesh
  - Utilizes the common low-level mesh interfaces
  - All TSTT mesh tools will be available
- Interface will be extensible, allowing user-defined operators and boundary conditions

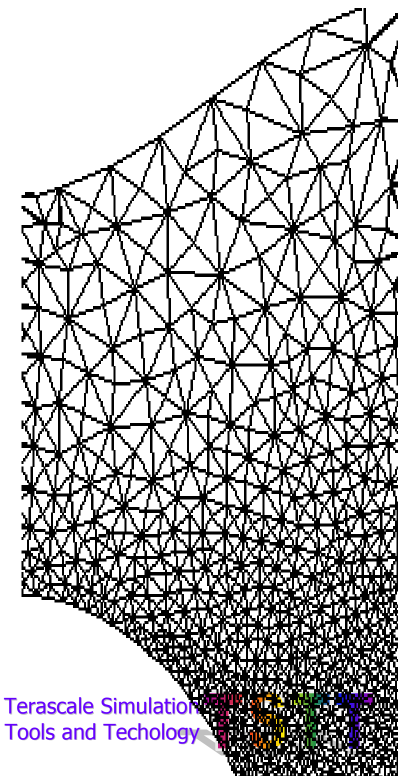
TSTT low-level common interfaces provide interoperable functionality for existing applications

- Examples:
  - Element-by-element access to mesh components
  - Fortran-callable routines that return interpolation coefficients at a single point (or array of points)
  - Fortran-callable routines that return metrics, derivatives, etc.



Mesh Component Interface Specification

- **Philosophy**
  - Create a small set of interfaces that existing packages can support
  - Balance performance and flexibility
  - Work with a large tool provider and application community to ensure applicability
- **Status**
  - Nomenclature is determined
  - Interfaces for basic access to mesh geometry and topology
  - Prototype implementations
    - All TSTT sites
    - C, C++, Fortran and SIDL
- **To Do**
  - Parallel query, mesh services, canonical orderings, mesh modification, structured mesh interfaces, tags…
  - Interaction with discretization strategies and field managers



Access to different meshing technology without knowledge of internal data structures is provided by TSTT Common Interface Specification

- Defined mesh entities and topologies
  - Vertex, Edge, Face and Region
  - Polygon, Triangle, Quadrilateral, Polyhedral, Tetrahedron, Hexahedron, Prism, Pyramid, Septahedron
- Opaque Objects
  - Mesh_Handle, Entity_Handle, Mesh_Error
- Functions
  - Mesh Create, Load, Services, GetEntities, GetEntityAdjacencies, Destroy
  - Entity Type, Topology, Dimension, Adjacencies, Vertex Coords
- Proposed Functions
  - Entity Interators, Tag add/get/set/delete, Services, Mesh Sets, Parallel Queries

Issues that have arisen

- Agreeing on nomenclature is harder than we first thought
- Cannot achieve the 100 percent solution, so...
  - What level of functionality should be supported?
    - Minimal interfaces only?
    - Interfaces for convenience and performance?
  - What about support of existing packages?
    - Are there atomic operations that all support?
    - What additional functionalities from existing packages should be required?
  - What about additional functionalities such as locking?
- Language interoperability is a problem
  - Most TSTT tools are in C++, most target applications are in Fortran
  - How can we avoid the "least common denominator" solution?
  - Exploring the SIDL/Babel language interoperability tool

Minimal Interfaces? … No

- Would imply an unstructured mesh access pattern
  - Node points and element connectivity
  - Inefficient (and unacceptable) to the structured grid developers/users
- Element-by-element access through function calls is too slow
- The least common denominator language solution (e.g. Fortran) is not acceptable to tool developers

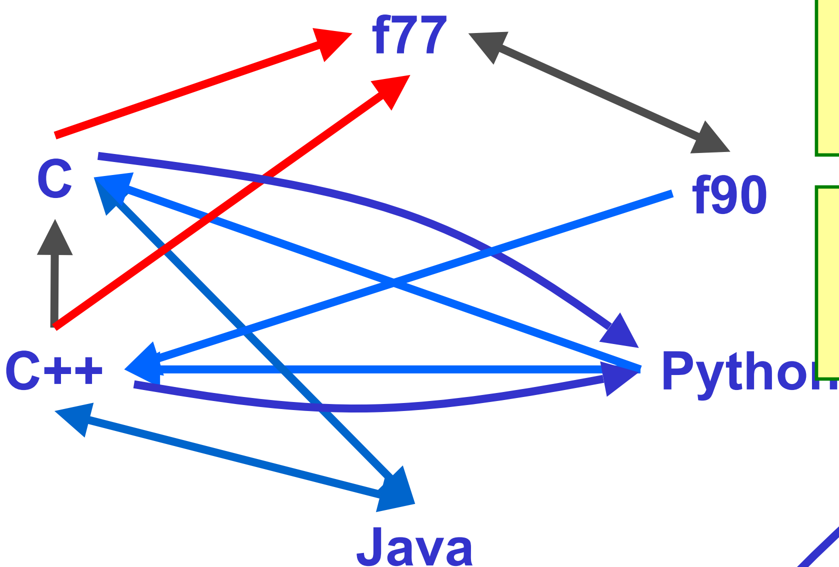
Interfaces for Performance and Flexibility?… Absolutely

- Interfaces for
  - Both structured and unstructured meshes
  - Both array and iterator access
  - Services to indicate what is available
- SIDL/Babel for language flexibility
  - Some language or package specific interfaces that can be compiled in if desired
- Worksets (Ranges)
  - Allow agglomerated access to reduce overhead

Are there atomic operations that all support? Well, yes, but…

- For example, all mesh packages *could* support element-by-element access, but it's not efficient
- Assume that all mesh packages support
  - Vertex coordinate information
- Require that all mesh packages support
  - Services to indicate what is supported
  - Stub implementations of the entire core specification
  - Eventually, some mechanism for associating field to mesh entities

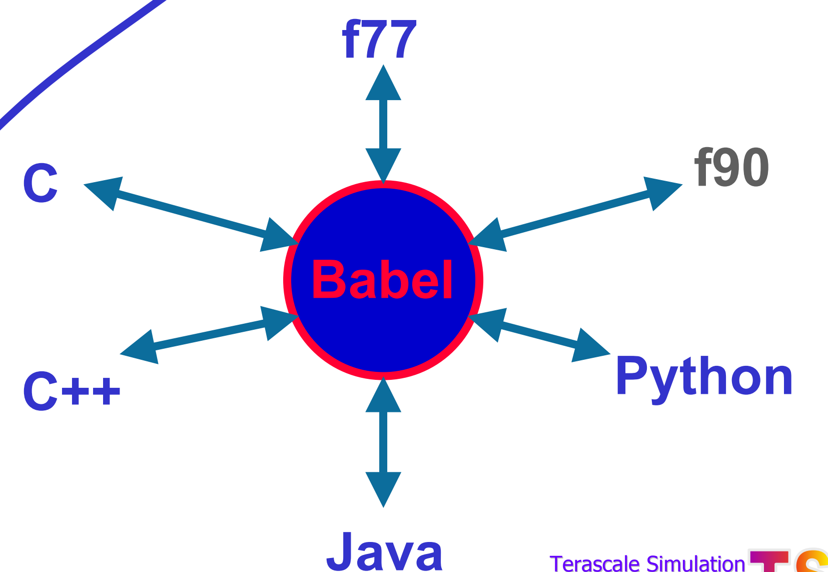
Component Technology ISIC's Babel provides a unified interoperability approach in which all languages are peers



Java JNI
Native
SWIG/SILOON
Platform Dependent
Python Library

Conventional approach requires hand-coding for each language

Babel replaces this with a single implementation using SIDL



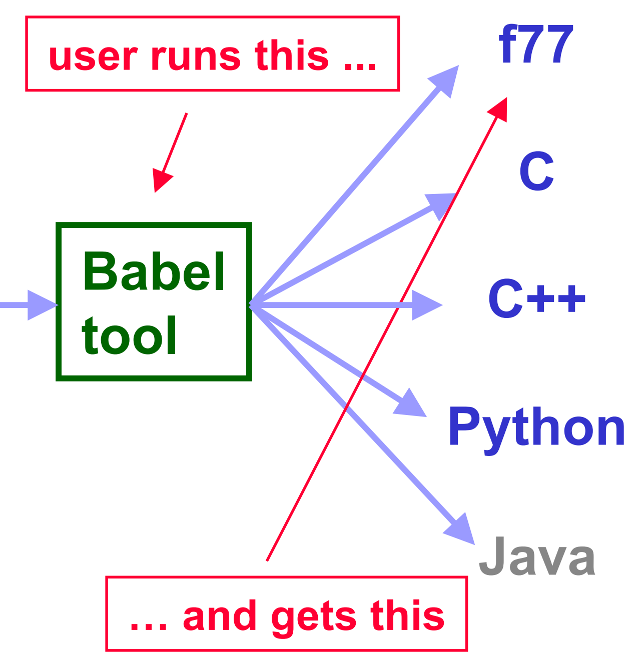
Babel uses a SIDL library description to generate glue code

- SIDL is a “scientific” interface definition language
  - Industry IDL approaches were modified for the scientific domain
  - SIDL describes calling interfaces (e.g., a library specification)
  - our tools automatically generate code to “glue languages”

TSTTMesh.sidl

```
version TSTTMesh 1.0;
package TSTTMesh {
   class Mesh {
      int getVertexCoords(in int ID, out array<double,1> x);
   }
}
```

library writer develops this



http://www.llnl.gov/CASC/components

More about TSTT Center activities later today in TSTT Sessions I, II and III

- MESQUITE: mesh quality improvement

- Improved accuracy for front tracking
- CAD to overlapping grids
- Higher order methods
- An implementation of the TSTT mesh interface (SNL)

- Rap: CAD repair tool
- Mesh distribution on parallel machines
- Grid generation tools for biology

Creating Interoperable Meshing and Discretization Technology:

The Terascale Simulation Tools and Technologies Center

David Brown *(Lawrence Livermore*)

Lori Freitag *(Argonne)*

Jim Glimm *(Brookhaven, SUNY Stony Brook)*

June 3, 2002

Terascale
Simulation
Tools
and
Techology